Mellanox OpenStack and SDN/OpenFlow Solution Reference Architecture

Rev 1.2

September 2013

www.mellanox.com

NOTE:

THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT (“PRODUCT(S)”) AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES “AS-IS” WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies, Ltd.
Beit Mellanox
PO Box 586 Yokneam 20692
Israel
www.mellanox.com
Tel: +972 (0)74 723 7200
Fax: +972 (0)4 959 3245

© Copyright 2013. Mellanox Technologies. All Rights Reserved.

Mellanox®, Mellanox logo, BridgeX®, ConnectX®, CORE-Direct®, InfiniBridge®, InfiniHost®, InfiniScale®, MLNX-OS®, PhyX®, SwitchX®, UFM®, Virtual Protocol Interconnect® and Voltaire® are registered trademarks of Mellanox Technologies, Ltd.

Connect-IB™, FabricIT™, Mellanox Open Ethernet™, Mellanox Virtual Modular Switch™, MetroX™, MetroDX™, ScalableHPC™, Unbreakable-Link™ are trademarks of Mellanox Technologies, Ltd.

All other trademarks are property of their respective owners

Contents

1 Solution Overview 7
1.1 OpenStack 7
1.2 Software Defined Networking (SDN) 9
1.2.1 OpenFlow 10
1.2.2 Supported Features 11
2 Accelerating Storage 12
3 Network Virtualization on Mellanox Adapters 14
3.1 Performance Measurements 15
3.2 Quality of Service Considerations 15
3.3 Seamless Integration 17
4 Setup and Installation 18
4.1 Basic Setup 18
4.2 Hardware Requirements 18
4.3 Software Requirements 19
4.4 Prerequisites 19
4.5 OpenStack Software installation 19
4.6 OpenFlow Agent Installation 20
5 Setting Up the Network 21
5.1 Configuration Examples 21
5.1.1 Creating a Network 21
5.1.2 Creating a Para-Virtualized vNIC Instance 22
5.1.3 Creating an SR-IOV Instance 23
5.1.4 Creating a Volume 24
5.1.5 Attach a Volume 25
5.2 Verification Examples 26
5.2.1 Instances Overview 26
5.2.2 Connectivity Check 26
5.2.3 Volume Check 26
5.3 OpenFlow Configuration Examples 27
5.3.1 Drop SSH Traffic from a Given Source IP Address 27
5.3.2 Set QoS Egress Queue 28

List of Figures

Figure 1: Mellanox OpenStack Architecture 8
Figure 2: OpenFlow Architecture 11
Figure 3: OpenStack Based IaaS Cloud POD Deployment Example 12
Figure 4: RDMA Acceleration 13
Figure 5: eSwitch Architecture 14
Figure 6: Latency Comparison 15
Figure 7: QoS, Setup Example 16
Figure 8: QoS, Test Results 16
Figure 9: Network Virtualization 17
Figure 10: Mellanox MCX314A-BCBT, ConnectX-3 40GbE Adapter 18
Figure 11: Mellanox SX1036, 36x40GbE 19
Figure 12: Mellanox 40GbE, QSFP Copper Cable 19
Figure 13: Quantum net-create/subnet-create Commands 21
Figure 14: OpenStack Dashboard, Instances 22
Figure 15: OpenStack Dashboard, Launch Instance 22
Figure 16: OpenStack Dashboard, Launch Interface – Select Network 23
Figure 17: Quantum port-create Command 23
Figure 18: Using the nova boot Command 24
Figure 19: OpenStack Dashboard, Volumes 24
Figure 20: OpenStack Dashboard, Create Volumes 24
Figure 21: OpenStack Dashboard, Volumes 25
Figure 22: OpenStack Dashboard, Manage Volume Attachments 25
Figure 23: VM Overview 26
Figure 24: Remote Console Connectivity 26
Figure 25: OpenStack Dashboard, Volumes 26
Figure 26: OpenStack Dashboard, Console 27

Preface

About this Document

This reference design presents the value of using Mellanox interconnect products and describes how to integrate the OpenStack /OpenFlow solution with the end-to-end Mellanox interconnect products.

Audience

This reference design is intended for server and network administrators.

The reader must have experience with the basic OpenStack framework and installation.

References

For additional information, see the following documents:

Table 1: Related Documentation

Reference	Location
Mellanox OFED User Manual	www.mellanox.com > Products > Adapter IB/VPI SW > Linux SW/Drivers http://www.mellanox.com/content/pages.php?pg=products_dyn&product_family=26&menu_section=34
Mellanox software source packages	https://github.com/mellanox-openstack
OpenStack Website	www.openstack.org
Mellanox OpenStack wiki page	https://wiki.openstack.org/wiki/Mellanox-OpenStack
Mellanox approved cables	http://www.mellanox.com/related-docs/user_manuals/Mellanox_approved_cables.pdf
Mellanox Ethernet Switch Systems User Manual	http://www.mellanox.com/related-docs/user_manuals/SX10XX_User_Manual.pdf
Mellanox Ethernet adapter cards	http://www.mellanox.com/page/ethernet_cards_overview
Solutions space on Mellanox community	http://community.mellanox.com/community/support/solutions
OpenFlow RPM package	http://community.mellanox.com/docs/DOC-1188

Reference	Location
OpenStack RPM package	http://community.mellanox.com/docs/DOC-1187
Mellanox eSwitchd Installation for OpenFlow and OpenStack	http://community.mellanox.com/docs/DOC-1126
Troubleshooting	http://community.mellanox.com/docs/DOC-1127
Mellanox OFED Driver Installation and Configuration for SR-IOV	http://community.mellanox.com/docs/DOC-1317

1 Solution Overview

1.1 OpenStack

Deploying and maintaining a private or public cloud is a complex task, with various vendors developing tools to address the different aspects of the cloud infrastructure, management, automation, and security. These tools tend to be expensive and create integration challenges for customers when they combine parts from different vendors. Traditional offerings suggest deploying multiple network and storage adapters to run management, storage, services, and tenant networks. These also require multiple switches, cabling, and management infrastructure, which increases both up front and maintenance costs.

Other, more advanced offerings provide a unified adapter and first level ToR switch, but still run multiple and independent core fabrics. Such offerings tend to suffer from low throughput because they do not provide the aggregate capacity required at the edge or in the core; and because they deliver poor application performance due to network congestion and lack of proper traffic isolation.

Several open source “cloud operating system” initiatives have been introduced to the market, but none has gained sufficient momentum to succeed. Recently OpenStack has managed to establish itself as the leading open source cloud operating system, with wide support from major system vendors, OS vendors, and service providers. OpenStack allows central management and provisioning of compute, networking, and storage resources, with integration and adaptation layers allowing vendors and/or users to provide their own plug-ins and enhancements.

Mellanox Technologies offers seamless integration between its products and OpenStack layers and provides unique functionality that includes application and storage acceleration, network provisioning, automation, hardware-based security, and isolation. Furthermore, using Mellanox interconnect products allows cloud providers to save significant capital and operational expenses through network and I/O consolidation and by increasing the number of virtual machines (VMs) per server.

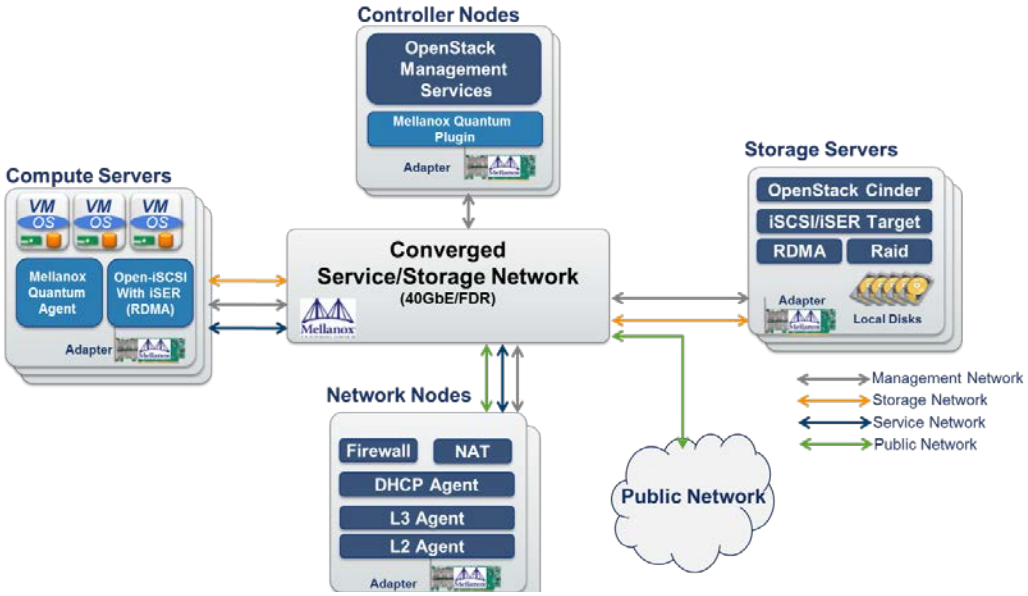
Mellanox provides a variety of network interface cards (NICs) supporting one or two ports of 10GbE, 40GbE, or 56Gb/s InfiniBand. These adapters simultaneously run management, network, storage, messaging, and clustering traffic. Furthermore, these adapters create virtual domains within the network that deliver hardware-based isolation and prevent cross-domain traffic interference.

In addition, Mellanox Virtual Protocol Interconnect (VPI) switches deliver the industry’s most cost-effective and highest capacity switches (supporting up to 36 ports of 56Gb/s). When deploying large-scale, high-density infrastructures, leveraging Mellanox converged network VPI solutions translates into fewer switching elements, far fewer optical cables, and simpler network design.

Mellanox integration with OpenStack provides the following benefits:

- Cost-effective and scalable infrastructure that consolidates the network and storage to a highly efficient flat fabric, increases the VM density, commoditizes the storage infrastructure, and linearly scales to thousands of nodes
- Delivers the best application performance with hardware-based acceleration for messaging, network traffic, and storage
- Easy to manage via standard APIs. Native integration with OpenStack Quantum (network) and Cinder (storage) provisioning APIs
- Provides tenant and application security/isolation, end-to-end hardware-based traffic isolation, and security filtering

Figure 1: Mellanox OpenStack Architecture



Mellanox Technologies Confidential

1.2 Software Defined Networking (SDN)

Software Defined Networking (SDN) is emerging as an alternative to proprietary data center networks. SDN architecture separates the control plane from the data plane in data center switches and hosts. With SDN, network control is implemented in software and can be executed from a server, which reduces network complexity and provides a common interface as an alternative to the proprietary and expensive options from traditional vendors. At the basis of the SDN approach is the decoupling of the system that makes decisions as to where traffic is sent (the control plane) from the underlying system that forwards traffic to the selected destination (the data plane). This enables network architects programmatically deciding how traffic flows throughout the network and centralizing this logic into a programmable interface that can be extended and tailored to customer needs.

SDN Approach Benefits

- Efficient and flexible networks (tailored optimization)
- Quick time-to-market of new services
- Cost savings on hardware (simpler forwarding devices required)
- Ability to test and implement new routing protocols quickly

Mellanox Technologies has been implementing these concepts for over 10 years in its InfiniBand products, providing existing data centers with a mature infrastructure for flexible, scalable, and dynamic networks. Today, Mellanox takes the extensive knowledge gained from building hundreds of high performance and scalable InfiniBand networks, and provides SDN networks on Ethernet as well, utilizing state-of-the-art technologies such as OpenFlow-enabled NICs and switches and an open architecture.

Mellanox's solution for SDN networks is built as an open, industry-standard platform which can deliver a wide range of Network Applications.

Mellanox integration with SDN/OpenFlow provides the following benefits:

- **Maximum Performance**

  Mellanox embedded eSwitch technology on its 40GbE NICs, together with an OpenFlow agent, provides the scalability and performance required for SDN security solutions. The eSwitch functionality enables hypervisor-like functionality in hardware while connecting with SR-IOV. This allows the customer to benefit from both worlds: policy enforcement via OpenFlow protocol and SR-IOV accelerated performance mode. A VM can access the network directly and execute the desired policy at near-line-rate performance.
  See Network Virtualization on Mellanox Adapters in chapter 3 of this document for more information on eSwitch.

- **OpenStack and OpenFlow seamless integration**

  Mellanox implements advanced provisioning logic to translate cloud service-level definitions and requirements to network provisioning commands. Mellanox supports the latest OpenStack release and utilizes industry-standard protocol Quantum API to integrate the Open Stack cloud management platform and Mellanox network devices. Both OpenStack and OpenFlow applications use the same eSwitch component on the ConnectX-3 adapter.

- **Overlay networks support**

  Mellanox customers are now able to benefit from the VxLAN and NVGRE scalability performance without compromising on network performance. ConnectX-3 supports high-performance 10GbE and 40GbE VxLAN and NVGRE. ConnectX-3 Pro eliminates the VxLAN and NVGRE performance overheads. It dramatically reduces CPU overhead up to 80% and enables 10GbE and 40GbE configuration with no throughput penalty. For additional information, see the ConnectX-3 Pro page on the Mellanox website.

- **Partnerships**

  Mellanox SDN is already enhancing partner solutions by eliminating traditional performance and scalability limitations.

1.2.1 OpenFlow

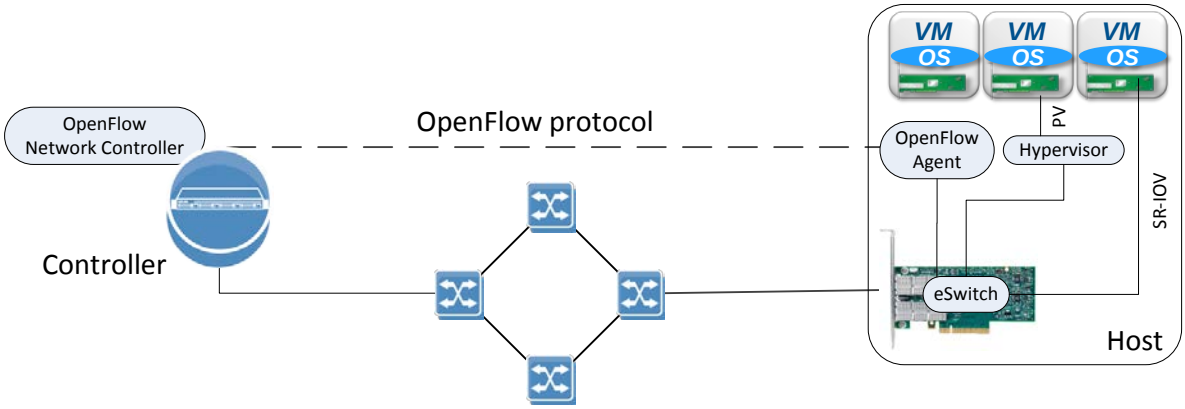
OpenFlow protocol provides a standard API between the control plane and forwarding plane.

ConnectX-3 incorporates an embedded switch (eSwitch) enabling VM communication to enjoy bare metal performance.

The ConnectX-3 driver includes OpenFlow agent software, based on the Indigo2 open source project, which enables controlling the eSwitch using standard OpenFlow protocol (the current OpenFlow version supported is 1.0).

Installing fabric flows on adapter eSwitches has great value and allows networks to scale naturally. Each eSwitch is responsible only for a relatively few VMs (only those VMs running on a specific host). Therefore, by distributing these switches on many adapters the scaling obstacle are eliminated. This is unlinke the case of trying to implement scalability on centralized physical switches which can support only relatively small numberof flows.

Figure 2: OpenFlow Architecture



In general, any OpenFlow controller (for example, Floodlight) can be used to interface the OpenFlow agent on ConnectX-3 adapters as long as the OpenFlow protocol versions are compatible.

1.2.2 Supported Features

The following OpenFlow Match fields are supported:

- Destination MAC address
- VLAN ID
- Ether Type
- Source/Destination IP address
- Source/Destination UDP/TCP port

Notes:

  - Field bitmask is not supported.
  - Destination MAC must be included.

The following OpenFlow Action fields are supported:

- Drop – providing security
- Set queue – providing QoS on fabric port, to which egress queue the flow is steered

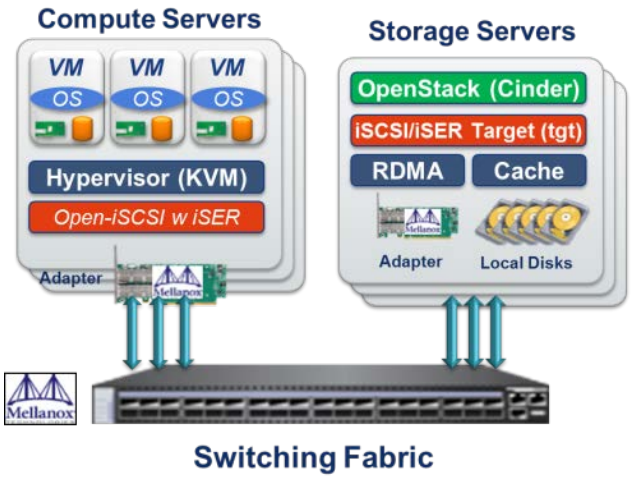
Flow Counters[1] are currently not supported.

[1] Roadmap

2 Accelerating Storage

Data centers rely on communication between compute and storage nodes, as compute servers read and write data from the storage servers constantly. In order to maximize the server's application performance, communication between the compute and storage nodes must have the lowest possible latency, highest possible bandwidth, and lowest CPU utilization.

Figure 3: OpenStack Based IaaS Cloud POD Deployment Example



Storage applications rely on iSCSI over TCP communications protocol stack continuously interrupt the processor in order to perform basic data movement tasks (packet sequence and reliability tests, re-ordering, acknowledgements, block level translations, memory buffer copying, etc). This causes data center applications that rely heavily on storage communication to suffer from reduced CPU efficiency, as the processor is busy sending data to and from the storage servers rather than performing application processing. The data path for applications and system processes must wait in line with protocols such as TCP, UDP, NFS, and iSCSI for their turn using the CPU. This not only slows down the network, but also uses system resources that could otherwise have been used for executing applications faster.

Mellanox OpenStack solution extends the Cinder project by adding iSCSI running over RDMA (iSER). Leveraging RDMA Mellanox OpenStack delivers 6X better data throughput (for example, increasing from 1GB/s to 5GB/s) and while simultaneously reducing CPU utilization by up to 80% (see Figure 4).

Mellanox ConnectX®-3 adapters bypass the operating system and CPU by using RDMA, allowing much more efficient data movement. iSER capabilities are used to accelerate hypervisor traffic, including storage access, VM migration, and data and VM replication. The use of RDMA shifts data movement processing to the Mellanox ConnectX-3 hardware, which provides zero-copy message transfers for SCSI packets to the application, producing significantly faster performance, lower network latency, lower access time, and lower CPU overhead. iSER can provide 6X faster performance than traditional TCP/IP based iSCSI. The iSER protocol unifies the software development efforts of both Ethernet and InfiniBand communities, and reduces the number of storage protocols a user must learn and maintain.

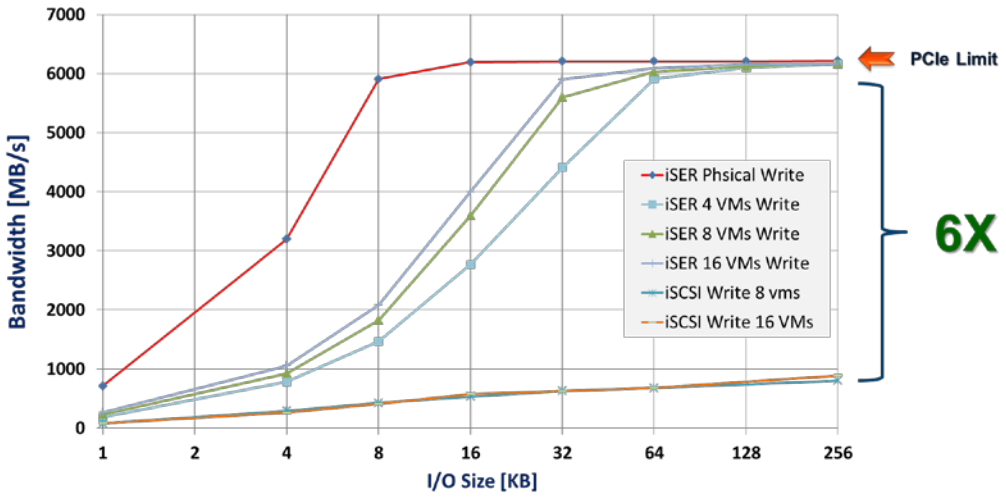
RDMA bypass allows the application data path to effectively skip to the front of the line. Data is provided directly to the application immediately upon receipt without being subject to various delays due to CPU load-dependent software queues. This has three effects:

- There is no waiting, which means that the latency of transactions is incredibly low.
- Because there is no contention for resources, the latency is deterministic, which is essential for offering end users a guaranteed SLA.
- Bypassing the OS, using RDMA results in significant savings in CPU cycles. With a more efficient system in place, those saved CPU cycles can be used to accelerate application performance.

In the following diagram, it is clear that by performing hardware offload of the data transfers using the iSER protocol, the full capacity of the link is utilized to the maximum of the PCIe limit.

To summarize, network performance is a significant element in the overall delivery of data center services and benefits from high speed interconnects.Unfortunately the high CPU overhead associated with traditional storage adapters prevents systems from taking full advantage of these high speed interconnects. The iSER protocol uses RDMA to shift data movement tasks to the network adapter and thus frees up CPU cycles that would otherwise be consumed executing traditional TCP and iSCSI protocols. Hence, using RDMA-based fast interconnects significantly increases data center application performance levels.

Figure 4: RDMA Acceleration



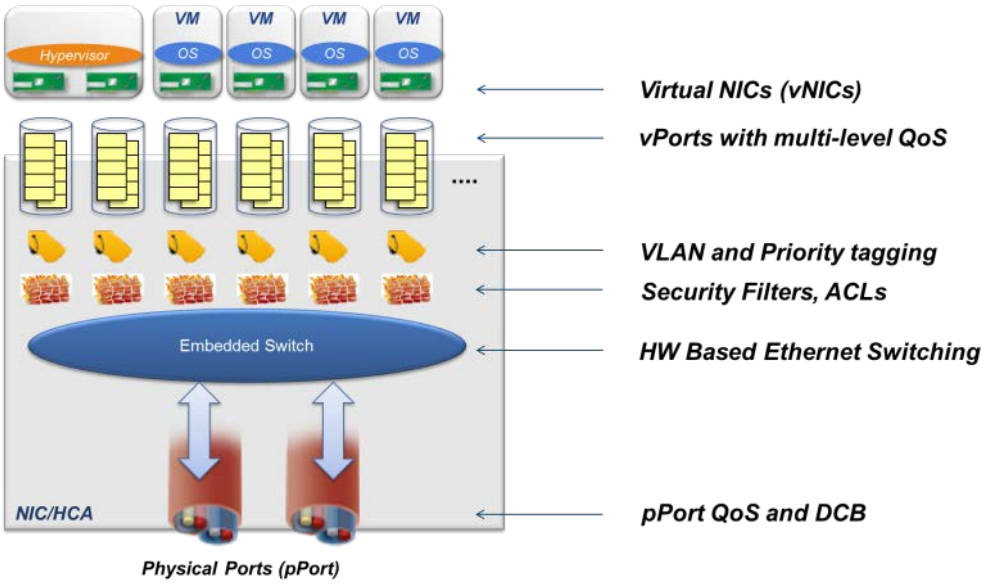
3 Network Virtualization on Mellanox Adapters

Single Root IO Virtualization (SR-IOV) allows a single physical PCIe device to present itself as multiple devices on the PCIe bus. Mellanox ConnectX®-3 adapters are capable of exposing up to 127 virtual instances called Virtual Functions (VFs). These virtual functions can then be provisioned separately. Each VF can be viewed as an additional device associated with the Physical Function. It shares the same resources with the Physical Function, and its number of ports equals those of the Physical Function.

SR-IOV is commonly used in conjunction with an SR-IOV enabled hypervisor to provide virtual machines with direct hardware access to network resources, thereby improving performance.

Mellanox ConnectX-3 adapters equipped with onboard embedded switch (eSwitch) are capable of performing layer-2 switching for the different VMs running on the server. Using the eSwitch will gain even higher performance levels and in addition improve security, isolation and QoS.

Figure 5: eSwitch Architecture



eSwitch main capabilities and characteristics:

- Virtual switching: creating multiple logical virtualized networks. The eSwitch offload engines handle all networking operations up to the VM, thereby dramatically reducing software overheads and costs.
- Performance: The switching is handled in hardware, as opposed to other applications that use a software-based switch. This enhances performance by reducing CPU overhead.

- Security: The eSwitch enables network isolation (using VLANs) and anti-MAC spoofing. In addition, by using OpenFlow ACLs, the eSwitch can be configured to filter undesired network flows.
- QoS: The eSwitch supports traffic class management, priority mapping, rate limiting, scheduling, and shaping configured via OpenFlow. In addition, DCBX control plane can set Priority Flow Control (PFC) and FC parameters on the physical port.
- Monitoring: Port counters are supported.

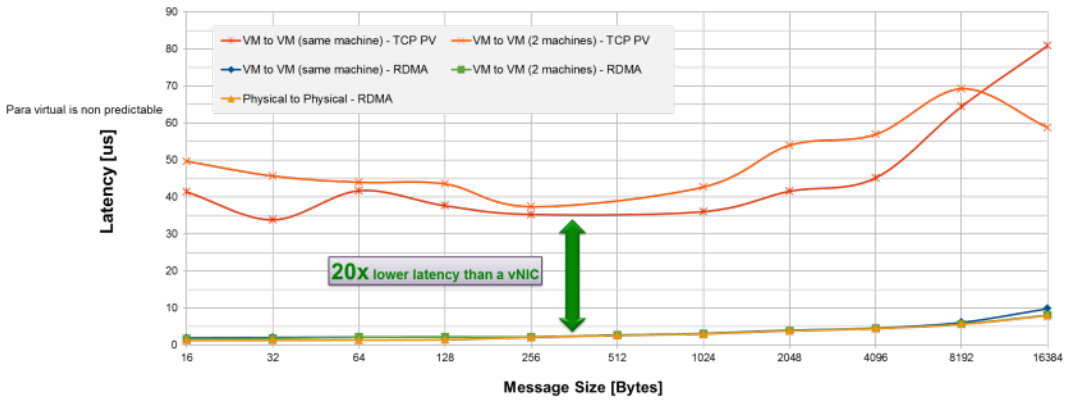
3.1 Performance Measurements

Many data center applications benefits from low latency network communication while others require deterministic latency. Using regular TCP connectivity between VMs can create high latency and unpredictable delay behavior.

Figure 6 shows the dramatic difference (20X improvement) delivered by SR-IOV connectivity running RDMA compared to para-virtualized vNIC running a TCP stream.

Using the direct connection of the SR-IOV and the ConnectX-3 hardware eliminates the software processing that adds an unpredictable delay to packet data movement. The result is a consistently low latency that allows application software to rely on deterministic packet transfer times.

Figure 6: Latency Comparison



3.2 Quality of Service Considerations

The impact of using QoS and network isolation is tremendous. The following example compares the various latency and bandwidth levels as a function of the QoS level.

The following test reveals the great advantage that can be achieved using the switch QoS capability:

Setup characteristics:

Streams:

In this test, two types of streams were injected:

- Blue (Storage stream): High bandwidth TCP stream. Latency is not crucial for such an application.
- Green (Messaging stream): Low bandwidth TCP messaging stream. Latency (as measured by Round Robin test) is crucial for such an application.

QoS levels:

The following QoS levels were tested:

- Single Queue: both streams use the same ingress queue.
- Dual Queues with no QoS: each stream has its own ingress queue while both queues have the same priority level.
- Dual Queues with QoS enabled: each stream has its own ingress queue while the green stream is prioritized over the blue stream.

Figure 7: QoS, Setup Example

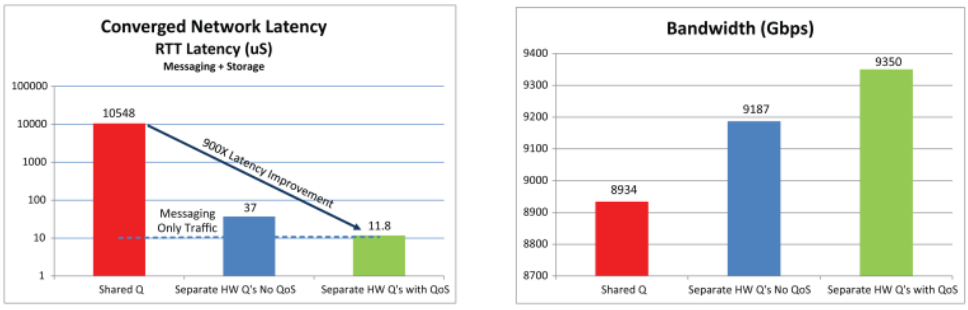
The test results show the following:

(1) When prioritizing a stream (green) and using dual queues, the low priority stream has a minor effect on the high priority stream (11.8µsec compared to 10.8µsec in Figure 8).

(2) Bandwidth increases when prioritizing streams (9350GbE), as well as when increasing the number of queues (9187GbE), compared to regular non-QoS conditions (8934GbE).

(3) The latency difference is dramatically reduced when using QoS (11.8µsec compared to 10548µsec).

Figure 8: QoS, Test Results



* Results are based on 10GbE adapter card

Conclusion:

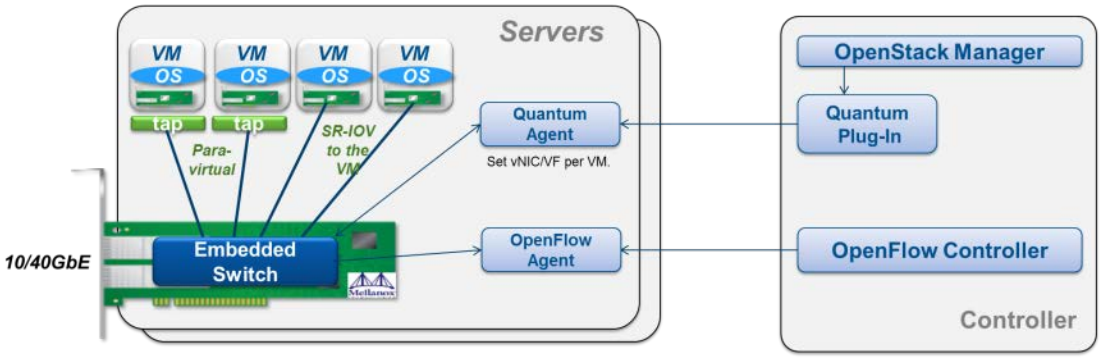
The test results emphasize that consolidation is possible on the same physical port. Applications that require low latency will not suffer from bandwidth-consuming applications when using more than one queue and enabling QoS.

3.3 Seamless Integration

The eSwitch configuration is transparent to the OpenStack or OpenFlow controller administrator. The installed eSwitch daemon on the server is responsible for hiding the low-level configuration. The administrator will use the standard OpenStack dashboard APIs or OpenFlow controller REST interface for the fabric management.

Both OpenFlow agent and Quantum agent configures the eSwitch in the adapter card.

Figure 9: Network Virtualization



4 Setup and Installation

4.1 Basic Setup

The following setup is suggested for small scale applications.

The OpenStack environment should be installed according to the OpenStack installation guide.

In addition, the following installation changes should be applied:

- A Quantum server should be installed with the Mellanox Quantum plugin.
- A Cinder patch should be applied to the storage servers (for iSER support).
- Mellanox Quantum agent, eSwitch daemon, and Nova patches should be installed on the compute notes.

4.2 Hardware Requirements

- Mellanox ConnectX®-3 adapter cards
- 10GbE or 40GbE Ethernet switches
- Cables required for the ConnectX-3 card (typically using SFP+ connectors for 10GbE or QSFP connectors for 40GbE)
- Server nodes should comply with OpenStack requirements.
- Compute nodes should have SR-IOV capability (BIOS and OS support).

There are many options in terms of adapters, cables, and switches. See www.mellanox.com for additional options.

Figure 10: Mellanox MCX314A-BCBT, ConnectX-3 40GbE Adapter

Figure 11: Mellanox SX1036, 36x40GbE

Figure 12: Mellanox 40GbE, QSFP Copper Cable

4.3 Software Requirements

- Supported OS
- RHEL 6.3 or higher
- Mellanox OFED 2.0.3 (SR-IOV support) or higher.
- KVM hypervisor – complying with OpenStack requirements

4.4 Prerequisites

(1) The basic setup is physically connected.

- In order to reduce the number of ports in the network, two different networks can be mapped to the same physical interface on two different VLANs.

(2) Mellanox OFED 2.0 (SR-IOV enabled) is installed on each of the network adapters.

- For Mellanox OFED installation refer to Mellanox OFED User Manual (Installation chapter).
  http://www.mellanox.com/page/products_dyn?product_family=26
- See Mellanox Community –for verification options and adaptation.
  http://community.mellanox.com/docs/DOC-1317

(3) The OpenStack packages are installed on all network elements.

(4) EPEL repository is enabled. (http://fedoraproject.org/wiki/EPEL).

4.5 OpenStack Software installation

For Mellanox OpenStack installation, follow the Mellanox OpenStack wiki pages:

- Quantum: https://wiki.openstack.org/wiki/Mellanox-Quantum
- Cinder: https://wiki.openstack.org/wiki/Mellanox-Cinder

For the eSwitch daemon installation, follow the OpenStack wiki pages (part of Mellanox Quantum):

- https://wiki.openstack.org/wiki/Mellanox-Quantum

4.6 OpenFlow Agent Installation

The OpenFlow agent installation procedure is defined in the Mellanox Community (http://community.mellanox.com/docs/DOC-1126).

5 Setting Up the Network

5.1 Configuration Examples

Once the installation is completed, it is time to set up the network.

Setting up a network consists of the following steps:

(1) Creating a network

(2) Creating a VM instance. Two types of instances can be created:

a. Para-virtualized vNIC

b. SR-IOV direct path connection

(3) Creating a disk volume

(4) Binding the disk volume to the instance that was just created

5.1.1 Creating a Network

Use the `quantum net-create` and `quantum subnet-create` commands to create a new network and a subnet (‘net3’ in the example).

Figure 13: Quantum net-create/subnet-create Commands

```
root@xena016-2:~# quantum net-create net3
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 97127c33-0095-4776-abb5-20420987009a |
| name                      | net3                                 |
| provider:network_type     | vlan                                 |
| provider:physical_network | mlx1                                 |
| provider:segmentation_id  | 3                                    |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 9511bf012f00467481022e3235b5786f     |
+---------------------------+--------------------------------------+
root@xena016-2:~# quantum subnet-create --name usbnet_net3 net3 192.168.203.0/24
Created a new subnet:
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| allocation_pools | {"start": "192.168.203.2", "end": "192.168.203.254"} |
| cidr             | 192.168.203.0/24                                     |
| dns_nameservers  |                                                      |
| enable_dhcp      | True                                                 |
| gateway_ip       | 192.168.203.1                                        |
| host_routes      |                                                      |
| id               | e0896b22-3668-4fce-bf8c-a843001858d5                 |
| ip_version       | 4                                                    |
| name             | usbnet_net3                                          |
| network_id       | 97127c33-0095-4776-abb5-20420987009a                 |
| tenant_id        | 9511bf012f00467481022e3235b5786f                     |
+------------------+------------------------------------------------------+
```

5.1.2 Creating a Para-Virtualized vNIC Instance

(1) Using the OpenStack Dashboard, launch an instance (VM) using the Launch Instance button.

(2) Insert all the required parameters and click Launch.

This operation will create a macvtap interface on top of a Virtual Function (VF).

Figure 14: OpenStack Dashboard, Instances

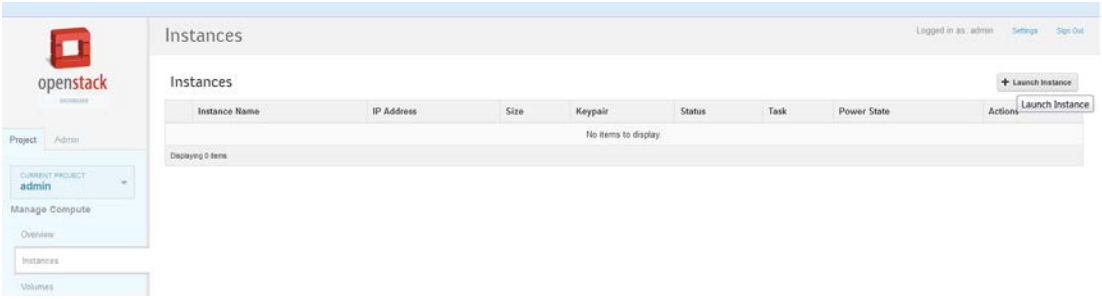
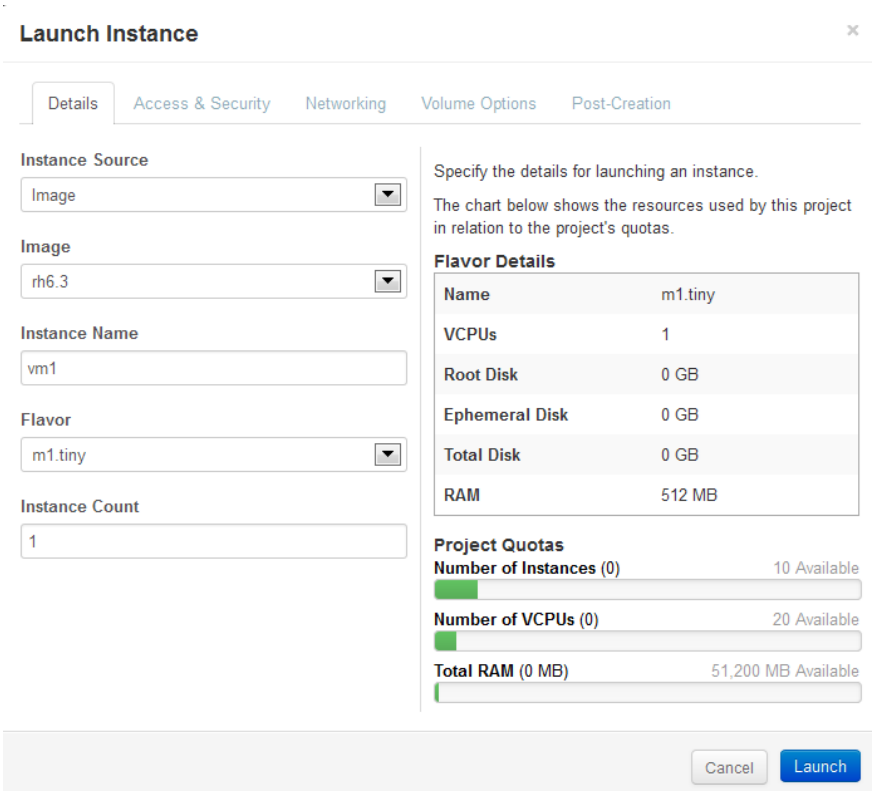
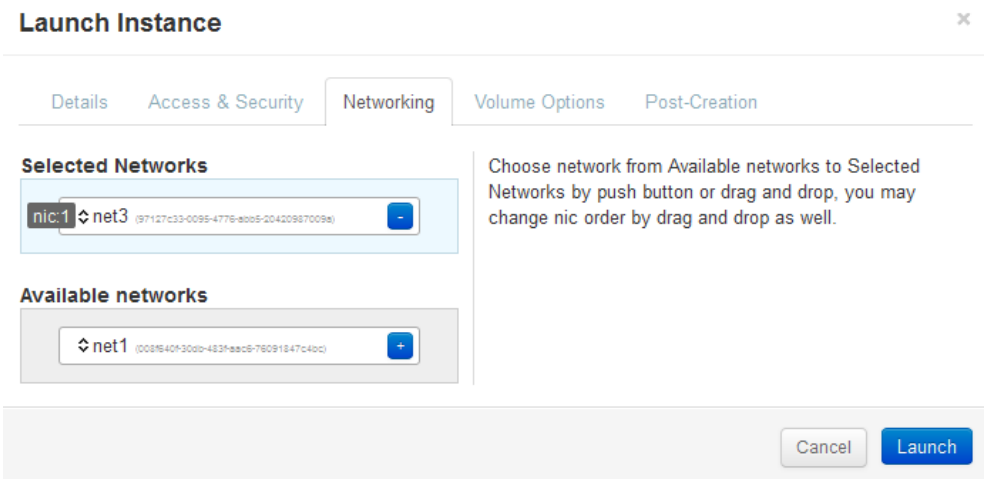


Figure 15: OpenStack Dashboard, Launch Instance



(3) Select the desired network for the vNIC (‘net3’ in the example).

Figure 16: OpenStack Dashboard, Launch Interface – Select Network



5.1.3 Creating an SR-IOV Instance

1. Use the `quantum port-create` command for the selected network (‘net3’ in the example) to create a port with ‘vnic_type=hostdev’.

Figure 17: Quantum port-create Command

```
root@xena016-2:~# quantum port-create net3 --binding:profile type=dict vnic_type=hostdev
Created a new port:
+---------------------+-------------------------------------------------------------------------------------+
| Field               | Value                                                                               |
+---------------------+-------------------------------------------------------------------------------------+
| admin_state_up      | True                                                                                |
| binding:capabilities | {"port_filter": false}                                                              |
| binding:profile     | {"physical_network": "mlx1"}                                                        |
| binding:vif_type    | hostdev                                                                             |
| device_id           |                                                                                     |
| device_owner        |                                                                                     |
| fixed_ips           | {"subnet_id": "e0896b22-3668-4fce-bf8c-a843001858d5", "ip_address": "192.168.203.5"} |
| id                  | 099a1db7-8b0a-4a7e-8045-2fd4cff4c17f                                                |
| mac_address         | fa:16:3e:b2:f1:6f                                                                   |
| name                |                                                                                     |
| network_id          | 97127c33-0095-4776-abb5-20420987009a                                                |
| status              | ACTIVE                                                                              |
| tenant_id           | 9511bf012f00467481022e3235b5786f                                                    |
+---------------------+-------------------------------------------------------------------------------------+
```

2. Use the nova boot command to launch an instance with the created port attached.

Figure 18: Using the nova boot Command

```
root@xena016-2:~# nova boot  --flavor m1.tiny --image rh6.3 vm3 --nic port-id=099a1db7-8b0a-4a7e-8045-2fd4cff4c17f
+-------------------------------------+--------------------------------------+
| Property                            | Value                                |
+-------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state               | scheduling                           |
| image                               | rh6.3                                |
| OS-EXT-STS:vm_state                 | building                             |
| OS-EXT-SRV-ATTR:instance_name       | instance-00000029                    |
| flavor                              | m1.tiny                              |
| id                                  | f2b0d3ec-64b8-473b-866a-b29264f13219 |
| security_groups                     | [{u'name': u'default'}]              |
| user_id                             | 2573e2ffdef241c29c932cc76f01e583     |
| OS-DCF:diskConfig                   | MANUAL                               |
| accessIPv4                          |                                      |
| accessIPv6                          |                                      |
| progress                            | 0                                    |
| OS-EXT-STS:power_state              | 0                                    |
| OS-EXT-AZ:availability_zone         | None                                 |
| config_drive                        |                                      |
| status                              | BUILD                                |
| updated                             | 2013-04-04T11:23:15Z                 |
| hostId                              |                                      |
| OS-EXT-SRV-ATTR:host                | None                                 |
| key_name                            | None                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname | None                                 |
| name                                | vm3                                  |
| adminPass                           | Uz3RPE3TvnJf                         |
| tenant_id                           | 9511bf012f00467481022e3235b5786f     |
| created                             | 2013-04-04T11:23:15Z                 |
| metadata                            | {}                                   |
+-------------------------------------+--------------------------------------+
```

5.1.4 Creating a Volume

Create a volume using the Volumes tab on the OpenStack dashboard. Click the Create Volume button.

Figure 19: OpenStack Dashboard, Volumes

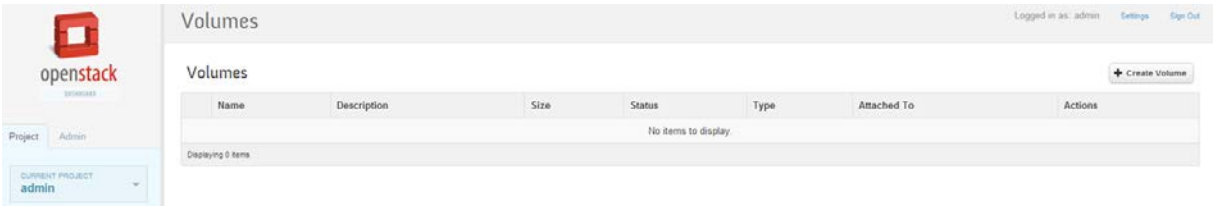


Figure 20: OpenStack Dashboard, Create Volumes

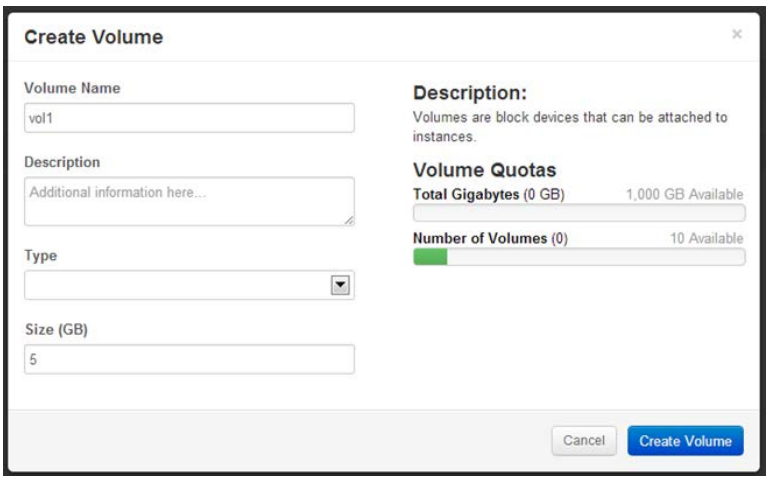
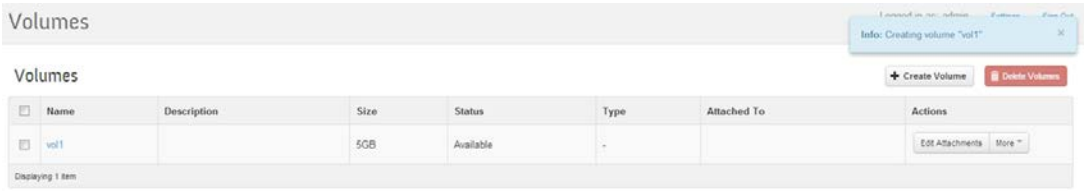


Figure 21: OpenStack Dashboard, Volumes

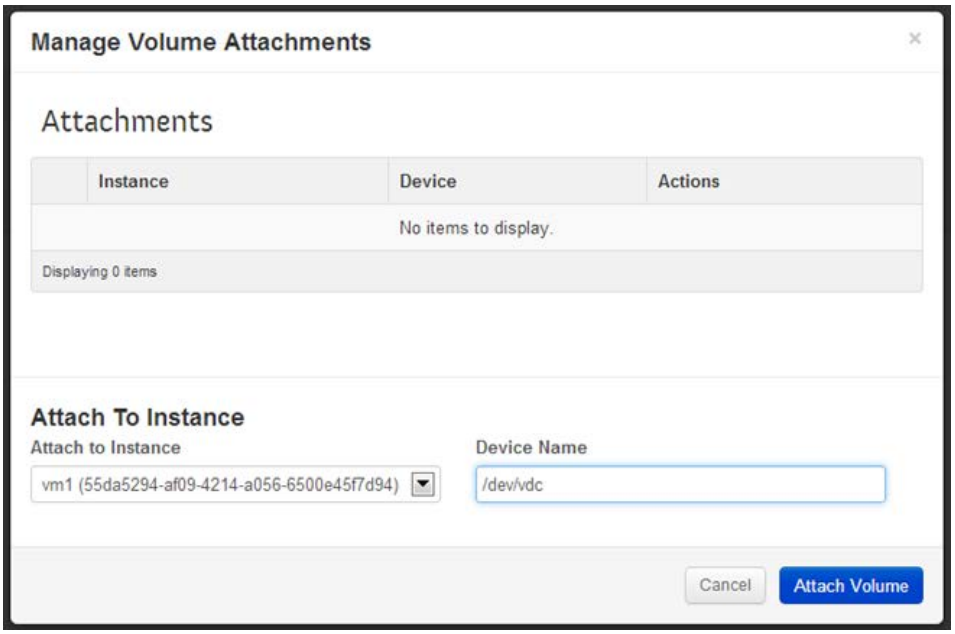


5.1.5 Attach a Volume

Attach a volume to the desired instance.

The device name should be /dev/dv<letter>. E.g. “/dev/vdc”

Figure 22: OpenStack Dashboard, Manage Volume Attachments



5.2 Verification Examples

5.2.1 Instances Overview

Use the OpenStack Dashboard to view all configured instances.

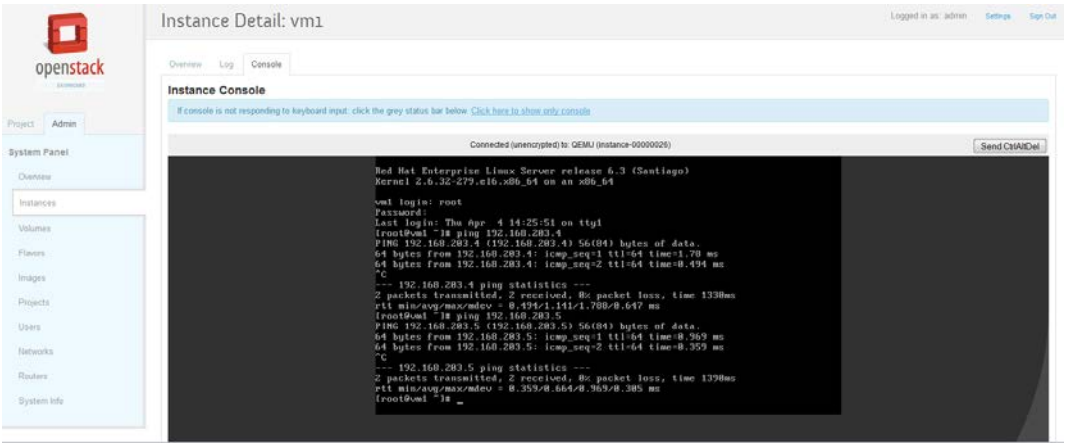
Figure 23: VM Overview

5.2.2 Connectivity Check

There are many options for checking connectivity between difference instances, one of which is simply to open a remote console and ping the required host.

To launch a remote console for a specific instance, select the Console tab and launch the console.

Figure 24: Remote Console Connectivity


openstack
Instance Detail: vm1
Overview
Log
Console
Instance Console
Send CtrlAltDel
Red Hat Enterprise Linux Server release 6.3 (Santiago)
Kernel 2.6.32-279.el6.x86_64 on an x86_64
vm1 login: root
Password:
Last login: Thu Apr 4 14:25:51 on tty1
[root@vm1 ~]# ping 192.168.203.4
PING 192.168.203.4 (192.168.203.4) 56(84) bytes of data.
64 bytes from 192.168.203.4: icmp_seq=1 ttl=64 time=1.70 ms
64 bytes from 192.168.203.4: icmp_seq=2 ttl=64 time=0.494 ms
^C
--- 192.168.203.4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1338ms
rtt min/avg/max/mdev = 0.494/1.141/1.788/0.647 ms
[root@vm1 ~]# ping 192.168.203.5
PING 192.168.203.5 (192.168.203.5) 56(84) bytes of data.
64 bytes from 192.168.203.5: icmp_seq=1 ttl=64 time=0.969 ms
64 bytes from 192.168.203.5: icmp_seq=2 ttl=64 time=0.359 ms
^C
--- 192.168.203.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1398ms
rtt min/avg/max/mdev = 0.359/0.664/0.969/0.305 ms
[root@vm1 ~]#

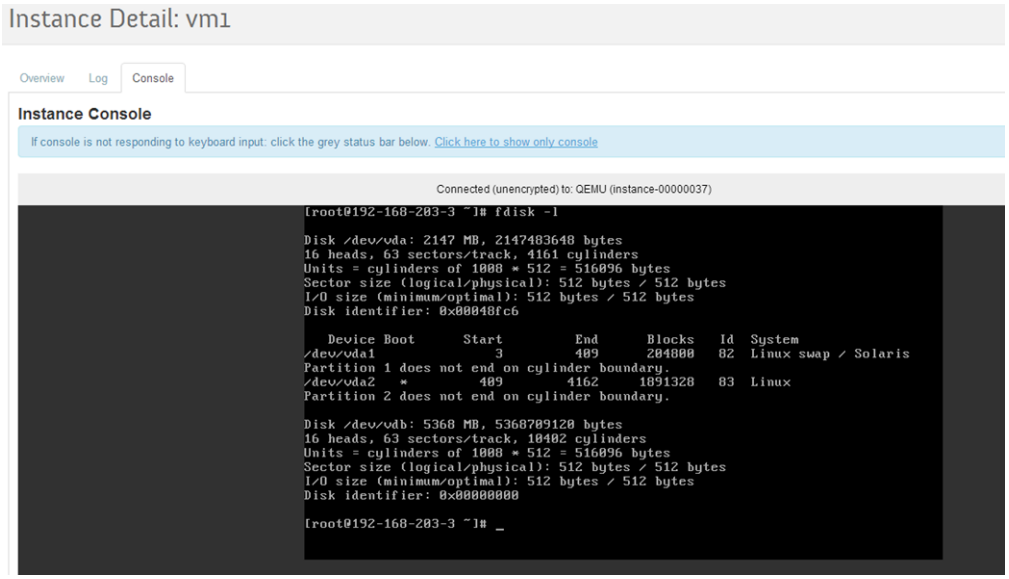
5.2.3 Volume Check

To verify that the created volume is attached to a specific instance, click the Volumes tab.

Figure 25: OpenStack Dashboard, Volumes

Additionally, run the `fdisk` command from the instance console to see the volume details.

Figure 26: OpenStack Dashboard, Console


Instance Detail: vm1
Overview
Log
Console
Instance Console
If console is not responding to keyboard input: click the grey status bar below. Click here to show only console
Connected (unencrypted) to: QEMU (instance-00000037)
[root@192-168-203-3 ~]# fdisk -l
Disk /dev/vda: 2147 MB, 2147483648 bytes
16 heads, 63 sectors/track, 4161 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00048fc6
Device Boot Start End Blocks Id System
/dev/vda1 3 409 204800 82 Linux swap / Solaris
Partition 1 does not end on cylinder boundary.
/dev/vda2 * 409 4162 1891328 83 Linux
Partition 2 does not end on cylinder boundary.
Disk /dev/vdb: 5368 MB, 5368709120 bytes
16 heads, 63 sectors/track, 10402 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
[root@192-168-203-3 ~]# _

5.3 OpenFlow Configuration Examples

The following examples use the Floodlight 0.9 network controller REST API.

Note: The flow must include a match on the destination MAC of the relevant VM's vNIC (the MAC that is assigned to the VF used by the VM).

5.3.1 Drop SSH Traffic from a Given Source IP Address

The following example shows how to apply a block rule for SSH traffic (TCP port 22) with source-ip IP 192.168.100.1 destined to a Virtual Machine with vNIC MAC 52:54:00:12:83:8e.

The OF_DPID is the identifier of the OpenFlow agent responsible for the HCA/eSwitch.

Perform the following commands on the controller:

MAC=52:54:00:12:83:8e

DST_PORT="22"

OF_IP="172.30.40.171"

OF_DPID="00:00:aa:bb:cc:dd"

SRC_IP="192.168.100.1"

curl -d '{"switch": """${OF_DPID}""", "name":"BLOCK-SSH-sw1", "cookie":"0", "priority":"32768", "dst-mac":"""${MAC}""", "ether-type":"2048", "src-ip":"""${SRC_IP}""",

"protocol":"6", "dst-port":"\"${DST_PORT}\"", "active":"true"}'
http://${OF_IP}:8080/wm/staticflowentrypusher/json

{"status" : "Entry pushed"}

Examine the rules on the HCA by running:

#ethtool -u eth4

4 RX rings available

Total 1 rules

Filter: 1

Rule Type: TCP over IPv4

Src IP addr: 192.168.100.1 mask: 0.0.0.0

Dest IP addr: 0.0.0.0 mask: 255.255.255.255

TOS: 0x0 mask: 0xff

Src port: 0 mask: 0xffff

Dest port: 22 mask: 0x0

Dest MAC addr: 52:54:00:12:83:8E mask: FF:FF:FF:FF:FF:FF

Action: Drop

Verify the configuration:

Try connecting to the server (from 192.168.100.1) via SSH – the operation should be denied.

Try and ping the server (from 192.168.100.1) – the operation should succeed.

5.3.2 Set QoS Egress Queue

The following example steers all traffic from the specific vNIC (specified by source MAC address 00:11:22:33:44:55) to egress queue 5.

curl -d '{"switch": "00:00:00:00:de:ad:be:ef", "name":"set-egress-queue-5", "cookie":"0", "priority":"32768", "src-mac":"00:11:22:33:44:55", "active":"true", "actions": "enqueue=0:5" }' http://172.30.49.68:8080/wm/staticflowentrypusher/json